Data Engineering – Project 5: Time Series

May 15, 2025

1 Introduction

Today's projects are all about time series data.

As usual, they exercises require that you load a dataset and process it as required, saving the indicated file. Please remember that the provided example datasets and configuration may be different from what is used to test your programs, but will always follow the guidelines specified in the exercises

2 Exercises

The file `proj5_params.json` contains a JSON dictionary with parameters which will be used throughout the exercises. Load it.

2.1 Exercise 1: Preparing time series datasets

4 points

The file `proj5_timeseries.csv` contains time series data. The first column contains timestamps, and the subsequent columns contain cumulative values for various parameters.

Please treat note that since the values are *cumulative*, they assume that they contain values aggregated over specific periods of time. For instance, for energy consumption, they will reflect the amounts of energy (expressed e.g. in Watt-hours) instead of *momentary* values (which are common in other fields of application, such as air quality parameters).

Also, please note that here we are dealing with a rather easy case – the dataset is alread synchronised (i.e. the different parameters are recorded with the same timestamps) and the data looks clean.

Perform the usual cleaning of column names – make letters (A–Z) lowercase and replace any other characters with underscores.

Convert the values in the *first* column to pandas datetime objects and set the column as the index.

The `original_frequency` contains the appropriate frequency symbol for the test dataset. Set the appropriate frequency in your DataFrame's index.

Save the cleaned DataFrame to `proj5_ex01.pkl`.

2.2 Exercise 2: Frequency adjustment

3 points

Change your DataFrame's frequency to the one indicated by the `target_frequency` parameter. Save the resulting DataFrame to `proj5_ex02.pkl`.

Please note that adjusting the frequency only *reindexes* a DataFrame, so it will not perform any aggregation (when moving towards lower frequencies) or interpolation (when moving towards higher ones) – though it can fill missing values in this case (but you're not asked to do it in the exercise).

2.3 Exercise 3: Downsampling

4 points

Please note that our dataset contains *cumulative* values – e.g., if the original frequency is *daily* (`D`), it contains the amount of energy consumed and generated on *each* subsequent 24-hour period. Therefore, by just adjusting the *frequency* of our DataFrame, e.g. from `D` to `W`, we effectively removed around $\frac{6}{7}$ of our data and, even worse, the totals no longer reflect the actual total values. In other words, each row in the resulting dataframe still contains the amount of energy produced/consumed during a day – one day in each week. Even though the frequency is now `W`, the values are not *weekly* totals. This problem would not occur if our dataset contained *momentary* values (e.g. the power drawn at a certain point in time).

In such case, we should rather use resampling. Parameters `downsample_periods` and `downsample_units` contain the number of periods (e.g. 3) and the units in which the periods are expressed (e.g. `d` for days), respectively.

Downsample your DataFrame accordingly, using an aggregation function appropriate for *cumulative* data, and make sure that the aggregates for a given "window" are calculated only if all samples from that window are present (otherwise, just put `NaN` in).

Save the resulting DataFrame to `proj5_ex03.pkl`.

2.4 Exercise 4: Upsampling

4 points

Parameters `upsample_periods` and `upsample_units` contain the number of periods (e.g. 2) and the units in which the periods are expressed (e.g. `h` for hours), respectively.

Parameters `interpolation` and `interpolation_order` contain the interpolation type (e.g. `polynomial`) and order (e.g. 3).

Upsample your original DataFrame to the target frequency and perform appropriate interpolation.

Please note that when interpolating, the numeric values will remain unmodified. Therefore, after we upsample and interpolate a dataset from, say, a 1-day (24-hour) frequency to a 2-hour frequnecy, someone looking at our DataFrame may this time assume that the values refer to energy produced/consumed during each 2-hour period. Therefore, scale the values according to the ratio between the original frequency and the upsampled one (hint: `pd.Timedelta`).

Save the DataFrame to `proj5_ex04.pkl`.

2.5 Exercise 5: Reshaping & alignment

6 points

In Exercise 1, we mentioned that this was actualy an *easy* case – the different parameters were already synchronized.

In reality, e.g. when dealing with sensor networks, the values will often come in the "long" form, where readings from different devices are in different lines, and the timestamps are not synchronized.

Read the file `proj5_sensors.csv`. The file contains a date/time index and two columns:

- `device_id`, containing the device identifier (this can be a number or a string),
- `value`, containing the reading value.

Parameters `sensors_periods` and `sensors_units` contain the number of periods (e.g. 3) and the units in which the periods are expressed (e.g. `d` for days), respectively.

Transform the loaded DataFrame so that:

- the index contains timestamps according to the specified frequency (e.g., for `10s`, the subsequent rows should be 10 seconds apart),
- individual columns are created for all devices, with the `device_id` as the column name,
- reading values are obviously placed on the intersection of columns and timestamps,
- in case a value for a specific timestamp isn't available, the values should be interpolated using linear interpolation,
- only the rows containing readings from all sensors are kept, i.e. if the readings from different sensors start/finish at different times, only include the period during which the data from all sensors is available.

Save the new DataFrame to `proj5_ex05.pkl`.

3 Submit your solution

As usual, commit your program to your GitLab project repository.

Save it as `project05/project05.py`.